Approches probabilistes pour la planification de tâches de manipulation

A. Sahbani
LAAS-CNRS
7, avenue du Colonel-Roche
31077 Toulouse Cedex - France
asahbani@laas.fr

Résumé - *Le problème général de planification de tâches de manipulation est un problème combinatoirement plus complexe que celui de la planification de mouvement. Les méthodes probabilistes permettent de faire face à cette complexité. Le problème traité ne se limite pas à planifier un chemin sans collision pour le robot mais également à trouver le séquencement des sous-tâches de prise et de pose d'objets.*
Dans cet article, on commencera par présenter la formulation géométrique du problème retenue ainsi que les travaux similaires proposés dans la littérature. On présentera également les algorithmes qui ont été développés afin de résoudre une certaine classe de problèmes de planification de tâche de manipulation, ainsi que les approches qui ont été adoptées.

Mots clé - Planification de mouvement, méthodes probabilistes, tâches de manipulation.

1 Introduction

Le problème de la planification de mouvement ou de trajectoire pour un robot a été largement étudié ces dernières années. De nombreux algorithmes ont été proposés dans la littérature afin de résoudre ce problème. Toutefois, les méthodes probabilistes s'avèrent efficaces dès que le nombre de degrés de liberté du robot devient important. Ces méthodes permettent de contourner la complexité exponentielle de ce problème. Cependant, l'extension de ces méthodes aux problèmes de planification de tâches de manipulation n'est pas triviale.
Le problème de la planification de tâches de manipulation est combinatoirement plus complexe que celui de la planification du mouvement. De plus, la présence d'objets déplaçables a un impact considérable sur la planification de mouvement d'un robot. Par exemple, quand il n'existe pas de chemin faisable entre deux configurations pour une disposition donnée des objets déplaçables dans l'environnement, le robot peut dégager un passage en changeant l'emplacement de certains objets dans la scène et atteindre la configuration finale malgré l'obstruction initiale. Par ailleurs, le but à assigner au robot peut également être un certain arrangement désiré des objets déplaçables.
Le problème de planification de tâches de manipulation ne se limite pas donc au niveau de la planification de mouvements sans collision pour un robot, mais également le séquencement des différentes sous-tâches de prise et de pose d'objets. Un objet déplaçable ne peut pas changer de position sauf s'il est manipulé par un robot. Nos travaux portent sur le développement d'approches de type probabilistes pour la planification de tâches de manipulation.

2 Formulation géométrique

La formulation géométrique retenue est celle introduite par R. Alami, J.P. Laumond, et T. Siméon dans [2]. Elle sera rappelée dans cette section puisque nos travaux sont basés sur cette formulation.

On se place pour la suite dans un environnement $3D$ contenant trois types d'objets: les obstacles statiques, le robot et les objets déplaçables. Pour le robot et pour chaque objet déplaçable, on associe un espace de configuration. L'espace de configuration pour chaque objet déplaçable est un espace de dimension 6, alors que celui du robot est de dimension n, où n est le nombre de degrés de liberté du robot. CS est le produit cartésien de tous ces espaces. Par ailleurs, CS_{free} représente l'ensemble des configurations dans lequel les différents objets ne sont pas en collision. Un chemin de manipulation est un chemin contraint dans CS_{free}. Il existe deux types de contraintes:

- Les contraintes de placements d'objet déplaçables.
- Les contraintes de mouvements d'objet déplaçables.

2.1 Les contraintes de placements d'objets déplaçables

Les configurations de CS_{free} ne correspondent pas forcément à des configurations physiquement valides. En effet, un objet ne peut pas être considéré dans une configuration valide s'il flotte sans support dans l'air. Une configuration d'un objet déplaçable est considérée

comme valide si elle est stable (au sens physique du terme). Notons par *PLACEMENT* le sous ensemble de CS_{free} où tous les objets déplaçables sont dans des configurations stables. Ainsi, chaque objet déplaçable présent dans l'environnement a un ensemble de "pose" déterminé en fonction de la position des obstacles et des autres objets déplaçables.
Notons que le sous-espace *PLACEMENT* dépend fortement du contexte.

2.2 Les contraintes de mouvements d'objets déplaçables

Contrairement à un robot, un objet déplaçable ne peut pas bouger tout seul. Il ne peut être déplacé que par le robot. Pour déplacer un objet, le robot doit se mettre tout d'abord dans une position de "prise", puis activer l'opération d'attachement. Un attachement est une relation géométrique décrivant le placement de l'objet par rapport au robot. Notons par *GRASP* le sous ensemble de CS_{free} dans lequel le robot se déplace solidairement avec l'objet manipulé sans entrer en collision avec les obstacles ou avec les autres objets déplaçables.
La relation géométrique qui définie une prise est considérée comme une donnée de notre problème. Cependant, le planificateur doit déterminer la configuration adéquate du robot vérifiant cette relation entre l'organe terminal et l'objet à déplacer.

2.3 Quelques définitions

- **Définition1**
  On appelle *chemin de transit* tout chemin de CS_{free} dans lequel le robot se déplace tout seul sans collision. Les objets déplaçables sont supposés dans des positions stables. On peut en déduire donc qu'un *chemin de transit* est inclus dans le sous-espace *PLACEMENT*.

- **Définition2**
  On appelle *chemin de transfert* pour l'objet déplaçable M_i tout chemin de CS_{free} dans lequel le robot déplace M_i sans entrer en collision ni avec les obstacles ni avec les autres objets déplaçables. Ces derniers sont supposés stationnaires dans des configurations stables.

- **Définition3**
  Un *chemin de manipulation* est un chemin dans CS_{free} composé d'une séquence alternée de *chemins de transit* et de *chemins de transfert*. A chaque début (resp. fin) d'un *chemin de transfert*, le robot saisit (resp. pose) l'un des objet déplaçables.

2.4 Planifier des tâches de manipulation

En se basant sur les définitions introduites précédemment, on peut en déduire qu'un *chemin de transfert* et un *chemin de transit* sont connectés si et seulement si tous les deux ont des extrémités communes dans l'espace $PLACEMENT \cap GRASP$.
Le problème de planification de tâches de manipulation apparaît ainsi comme un problème de recherche de chemins dans les différentes composantes connexes du sous-espace $PLACEMENT \cap GRASP$ ou entre elles. L'exploration de l'espace $PLACEMENT \cap GRASP$ s'avère donc nécessaire pour la résolution d'un problème de planification de tâches de manipulation. On définit alors le *graphe de manipulation* comme étant le graphe dont les nœuds forment les composantes connexes de $PLACEMENT \cap GRASP$. Il existe deux types d'arêtes. Une arête de transfert (resp. de transit) liant deux nœuds indique qu'il existe un chemin de transfert (resp. de transit) connectant les deux configurations des composantes connexes associées.

3 Travaux similaires

Plusieurs algorithmes permettant la résolution de problème de planification de tâches de manipulation ont été proposés dans la littérature.
Alami, Laumond et Siméon présentent dans [2, 3] un schéma de résolution appliqué à deux instances de problèmes: cas de plusieurs objets manipulés par un polygone en translation et pour un bras à trois articulations rotoïdes sous l'hypothèse de nombre fini de prises et de poses, et cas d'un seul objet déplaçable dans un environnement polygonal avec un nombre fini de prises.
La solution de Koga et Latombe [6] consiste d'abord à planifier une trajectoire pour l'objet déplaçable, puis trouver une succession de *chemin de transfert* et de *chemin de transit* déplaçant l'objet le long de la trajectoire précalculée.
Ahuactzin et Mazer ont étendu l'algorithme de Fil d'Ariane afin de supporter des environnements dynamiques. L'extension est appelé "The Ariadne's clew algorithm" [4]. Il est basé sur des algorithmes génétiques. Il collecte des informations sur l'espace atteignable depuis la position initiale en plaçant des balises dans l'espace de recherche. En effet les balises représentent une configuration de prise stable atteignable à partir de la configuration initiale. Pour chaque nouvelle balise placée, le planificateur essaie d'atteindre la configuration but en exécutant un chemin local [1]. L'idée principale derrière cet algorithme est d'eviter de construire l'espace des configurations en conduisant les recherches dans l'espace des trajectoires.
Une autre méthode proposée par Nielsen et Kavraki est une extension des méthodes probabilistes (PRM: probabilistic roadmap approach). Cette méthode est appelée "Fuzzy PRM" [9]. Elle consiste à construire un graphe dont les arêtes sont annotées par une probabilité. Cette probabilité est une estimation de chance que l'arête est actuellement faisable (sans collision).
Après cette brève présentation des travaux similaires, les sections suivantes seront consacrées à la présenta-

tion des algorithmes que nous avons développés.

4 Résoudre un problème simple de manipulation

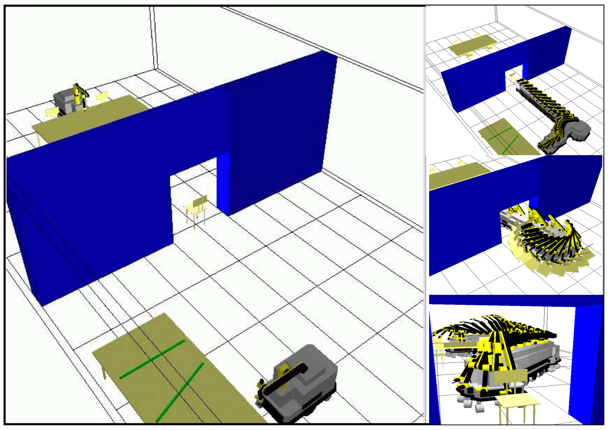

Figure 1: Exemple simple de prise et de pose.

Considérons le scénario illustré dans la Figure 1. Le robot partant de sa position initiale dans la première pièce, doit rejoindre sa position finale dans l'autre pièce. L'unique accès à la deuxième pièce est bloqué par une chaise.
Ce type de tâche peut être décomposé en trois sous-tâches: "*Va à la chaise*", "*Prend la chaise et pose la de façon à accéder à l'autre pièce*", "*Va à ta configuration but*". Chaque sous-tâche est traitée selon une approche probabiliste [11]. Ainsi, trois graphes sont calculés tenant compte de l'état courant de l'environnement. Le premier graphe (Figure 2(a)) représente le recouvrement de l'espace *GRASP*. Le graphe est constitué d'une seule composante connexe, ce qui implique que le robot peut poser la chaise là ou il veut, tout en respectant les contraintes de placements. Le deuxième graphe est calculé en considérant que la chaise occupe la configuration indiquée dans la Figure 2(b). Ce graphe recouvre donc, le sous-ensemble de *PLACEMENT* correspondant au placement initial de la chaise. Le troisième graphe présenté dans la Figure 2(c) couvre de même le sous-ensemble de *PLACEMENT* correspondant au placement final de la chaise. Le fait que dans le deuxième ainsi que dans le troisième cas, les graphes correspondant sont constitués de deux composantes connexes montrent que la chaise bloque bien le passage entre les deux pièces.

Dans ce type de problème, l'opérateur doit définir la position initiale et finale du robot ainsi que les différentes prises possibles des objets déplaçables. L'algorithme développé calcule les trois graphes décrits précédemment puis génére une trajectoire valide. Il est à noter qu'il faut rajouter aux graphes calculés des nœuds permettant la connexion et le passage d'un graphe à un autre. En outre, ces nœuds permettent la

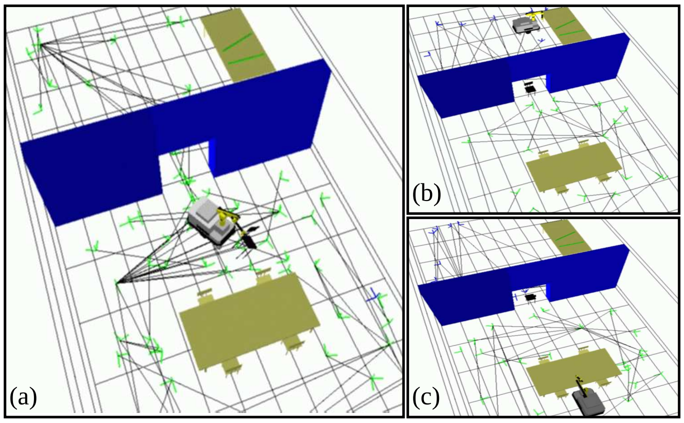

Figure 2: Les trois graphes calculés.

connexion entre *un chemin de transfert* et *un chemin de transit*.

5 Une nouvelle approche basée sur la notion de graphe dynamique

La génération d'un graphe est coûteuse en terme de temps de calcul. L'idée principale de cette approche est de minimiser le nombre de graphes calculés. Elle combine deux méthodes: réseaux probabiliste basés sur la visibilité [10, 11] et RRT (Rapidly-exploring Random Trees)[7, 8].

5.1 La méthode des réseaux probabilistes basés sur la visibilité

La méthode des réseaux probabilistes est une méthode de planification de mouvement pour des environnements statiques, complète en probabilité et qui n'est pas restreinte à une classe particulière de robots. Elle procède en deux phases: une phase d'apprentissage et une phase de recherche [5]. Lors de la phase d'apprentissage, un graphe est construit selon un tirage aléatoire dans l'espace des configurations. Les nœuds de ce graphe représentent des configurations libres (sans collision) du robot dans l'environnement. Les arêtes représentent des chemins faisables par le robot entre les configurations. Ces chemins sont construits selon la méthode locale adoptée. Après la construction du graphe, la phase de recherche est entamée. Etant données une configuration de départ et une configuration d'arrivée, on les relie au graphe par des chemins locaux faisables. Une fois liées, il nous reste qu'à chercher un chemin dans le graphe connectant les deux nœuds ajoutés.
La méthode des réseaux probabilistes basée sur la visibilité [10, 11] est une variante des méthodes probabilistes: un nœud n'est ajouté au graphe que s'il

n'appartient à aucun domaine de visibilité des nœuds déjà présent ou s'il permet de relier plusieurs composantes connexes du graphe.
Le premier avantage de l'approche est de définir un critère d'arrêt explicite de l'algorithme de construction du graphe, critère absent dans les approches jusqu'ici développées. Par ailleurs la taille réduite du graphe conduit à un gain significatif en temps de calcul.

5.2 RRT: méthode d'exploration rapide

Le formalisme d'exploration rapide et aléatoire des arbres a été introduit par LaValle dans [7, 8]. Cette méthode consiste a faire évoluer deux arbres ayant comme racine les deux configurations qu'on cherche à connecter. La construction des arbres se fait de façon incrémentale dans le but de réduire la distance d'un point tiré aléatoirement aux deux arbres. Les deux arbres progressent l'un vers l'autre selon une heuristique simple.
Cette méthode est adaptée au cas d'une recherche unique d'un chemin dans un environnement donné. Par ailleurs, cette méthode est complète en probabilité.

5.3 Notre approche

Au début, on calcule un graphe couvrant CS_{free} sans tenir compte de la présence d'objets déplaçables dans l'environnement. Afin de valider un *chemin de transit*, pour une répartition donnée des objets déplaçables dans l'environnement, le graphe est automatiquement mis à jour. Chaque nœud puis chaque arête sont testés afin de détecter les éventuelles collisions avec les positions courantes des objets déplaçables. Si un nœud est détecté en collision avec l'un des objets déplaçables, toutes les arêtes sortantes ou entrantes à ce nœud sont supprimées ainsi que le nœud correspondant. Cependant, si une arête est détectée en collision, sans que les nœuds (extrémités) ne le soient, Cette arête est marquée. La recherche de chemin se poursuit en utilisant le graphe ainsi marqué. Trois cas de figures peuvent se présenter. Le premier cas est de trouver un chemin ne comprenant aucune arête marquée, le chemin est donc validé et le problème est résolu. Le deuxième cas est de ne pas trouver un chemin, cela implique que le problème ne possède pas de solution. Finalement le dernier cas est lorsque le chemin contient au moins une arête marquée. Dans cette situation, l'algorithme essaie de résoudre le problème localement. La méthode RRT, décrite ci dessus, sert à remplacer le fragment en collision dans l'arête par un chemin tenant compte de la position actuelle des différents objets déplaçables. La complétude en probabilité de notre approche découle directement du fait que les deux méthodes utilisées, à savoir visibilité PRM et RRT, sont complètes en probabilité.
La même approche est utilisée pour la recherche d'un *chemin de transfert*. Le graphe initialement calculé est mis à jour en considérant qu'il y a un objet déplaçable attaché au robot et en tenant compte de la position actuelle des autres objets déplaçables (cf. Figure 3).

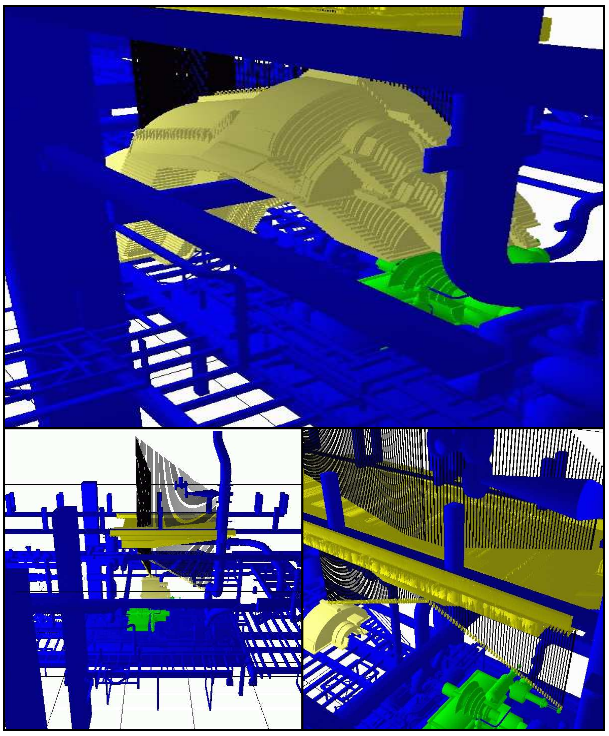

Figure 3: Transport d'une partie de turbo-pompe d'une centrale nucléaire (modèle EDF).

6 Extension au cas de plusieurs objets déplaçables

L'approche présentée dans cette section est une extension de l'approche précédente. On considère qu'il existe plusieurs objets déplaçables dans l'environnement. Pour chaque objet, on définit les différentes prises et poses possibles entant qu'entrée du problème. L'algorithme opère, comme les méthodes probabilistes, en deux phases: une phase d'apprentissage et une phase de recherche.
Lors de la phase d'apprentissage l'algorithme construit une liste de graphes. Le premier graphe est calculé pour le robot seul dans l'environnement sans tenir compte de la présence des objets déplaçables. Les autres graphes sont calculés pour chacune des positions de prise définies initialement. En effet, pour chaque relation géométrique définissant une prise, on construit un graphe pour le robot tenant l'objet déplaçable selon cette relation. Les objets déplaçables omis celui pris par le robot ne sont pas comptabilisés dans la scène lors du calcul des graphes. Les graphes sont calculés en utilisant la méthode des réseaux probabilistes basés sur la visibilité présenté dans la section 5.1.
Une fois la phase d'apprentissage achevée, la phase de recherche est lancée. Le graphe du robot seul est

enrichi par des nœuds spécifiques, si ces derniers ne figurent pas dans les graphes. Ces nœuds correspondent aux positions de prises et de poses pour chaque objet déplaçable. D'autre part, pour chaque graphe relatif au robot tenant un objet déplaçable, on rajoute les nœuds de prise et de poses correspondant à la prise courante.

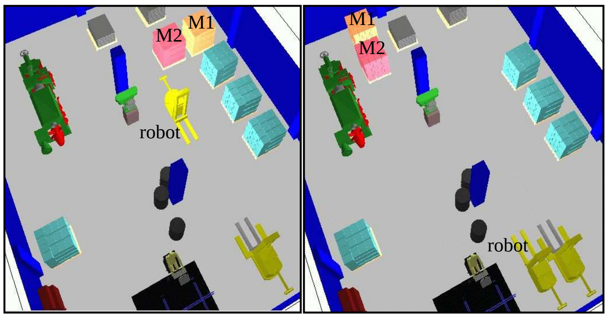

Figure 4: La position initiale et finale des deux objets à déplacer ainsi que celle du robot.

Considérons le scénario illustré dans la Figure 4. Le robot partant de sa position initiale doit aller chercher les deux objets déplaçables ($M1$, $M2$) l'un après l'autre et les mettre dont leurs positions finales. Afin d'expliquer cette approche, on se placera pour cet exemple dans le cas d'une configuration de prise possible et un ensemble fini de pose. La première phase consiste à calculer trois graphes. Le premier graphe ($G1$) est calculé pour le robot tout seul sans présence de $M1$ et $M2$. Les deux autres graphes ($G1\_1, G1\_2$) sont calculés pour le robot tenant un objet déplaçable, l'autre est considéré absent de la scène. Une fois les graphes générés, l'algorithme commence par ajouter le nœud au graphe $G1$ tenant compte de la position actuelle des deux objets déplaçables dans l'environnement. Ce nœud correspond à la configuration de prise de $M1$. Dans notre cas ce nœud ne peut pas être ajouté, il n'est pas valide. On passe ainsi au deuxième lien introduit dans la liste des attachements définissant le problème de manipulation. Le nœud correspondant à ce lien est ajouté au graphe $G1$. L'algorithme planifie un chemin de transit connectant la position initiale du robot avec la configuration de prise de $M2$, en utilisant le graphe $G1$. Une fois le chemin trouvé, une recherche de chemin de transfert est lancée entre la configuration du prise (position initiale de $M2$) et la configuration de pose finale (position but de $M2$), en utilisant le graphe $G1\_2$. En effet, la configuration de prise appartient à l'intersection de l'espace $PLACEMENT$ avec la sous variété de l'espace $GRASP$ relatif à la prise de $M2$. De ce fait le passage d'un graphe à un autre est justifié. Après, on rajoute le nœud correspondant à la configuration de prise de l'objet $M1$ au graphe $G1$. Cette configuration est maintenant valide puisque $M2$ a changé de position. L'algorithme cherche ainsi un chemin dans $G1$ connectant les deux configurations en question. La configuration de pose finale de l'objet $M1$ n'étant pas valide, l'algorithme cherche à connecter la configuration de prise actuelle à une configuration de pose valide parmi les poses définies dans la liste. En continuant selon la même méthode, l'algorithme continue incrémentalement à trouver une séquence de chemin de transfert et de chemin de transit pour résoudre ce problème de planification de tâche de manipulation. La séquence solution est présentée dans la figure 5.

Partant de sa position initiale le robot déplace l'objet $M2$ à sa position finale et revient chercher l'objet $M1$ (Figures 5.1, 5.2 et 5.3). La configuration du robot avec l'objet $M1$, placé dans sa position finale n'étant pas valide, l'algorithme planifie une succession de chemin de transfert et de transit permettant de placer cet objet dans une position de pose prédifinie (Figures 5.4). Ensuite, il revient déplacé $M2$ dans une position de pose et revient chercher $M1$ pour le transporter jusqu'à sa position finale (Figures 5.5, 5.6, 5.7 et 5.8). Dans les figures 5.9 et 5.10, le robot déplace l'objet $M2$ à sa position de pose finale. Finalement, l'algorithme planifie un chemin de transit ramenant le robot à sa configuration finale (Figure 5.11).

Il est à noter que dans l'algorithme présenté, on recherche pas à trouver un séquencement optimal des sous-tâches afin de résoudre un problème de manipulation mais plutôt un chemin de manipulation faisable.

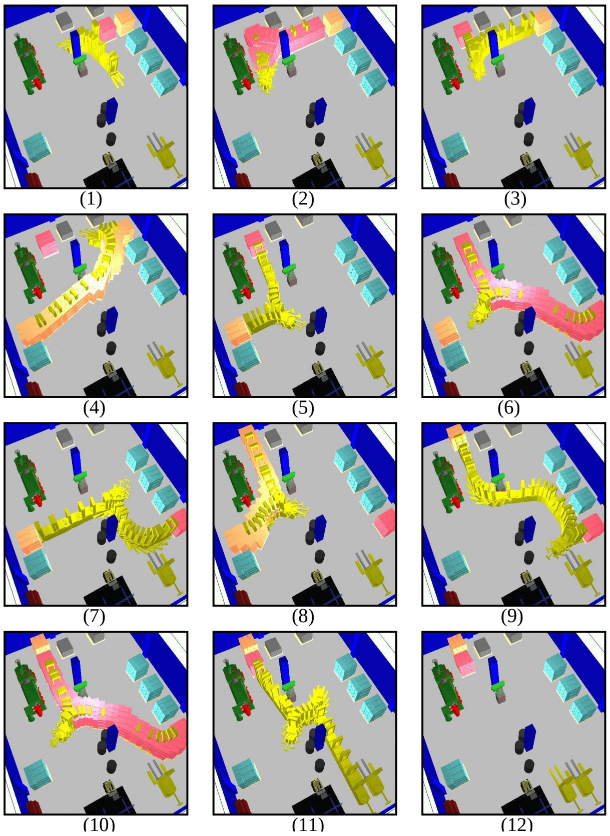

Figure 5: La séquence de chemin de transfert et de chemin de transit trouvée par le planificateur.

7 Exploration de l'espace *GRASP* ∩ *PLACEMENT*

A travers les approches introduites précédemment, l'exploration de l'espace $GRASP \cap PLACEMENT$ s'avèrent importante pour la résolution d'un problème de planification de tâches de manipulation. $GRASP \cap PLACEMENT$ est une variété de l'espace de configuration globale du robot mais ayant une dimension plus petite. La difficulté principale pour les méthodes probabilistes vient du fait que la probabilité de tirer aléatoirement une configuration dans cette variété est nulle.
L'idée principale pour l'exploration du sous-espace $PLACEMENT \cap GRASP$ est de considéré cet espace comme étant l'espace des configurations pour une seule chaîne cinématique. Cette chaîne est composée du robot avec l'objet déplaçable placé dans une position stable. Maintenir un placement stable de l'objet, lors du déplacement de l'ensemble implique la présence d'une chaîne cinématique fermée. Ceci est du au fait que l'objet déplaçable doit être continuellement en contact avec son support lors du déplacement de l'ensemble objet-robot. La figure 6 illustre le déplacement de l'ensemble tout en gardant le contact entre l'objet déplaçable et le sol. Ainsi, la topologie de l'espace $GRASP \cap PLACEMENT$ est capturée à travers un graphe calculé pour l'ensemble robot-objet avec la contrainte additionnelle concernant le placement de l'objet déplaçable.

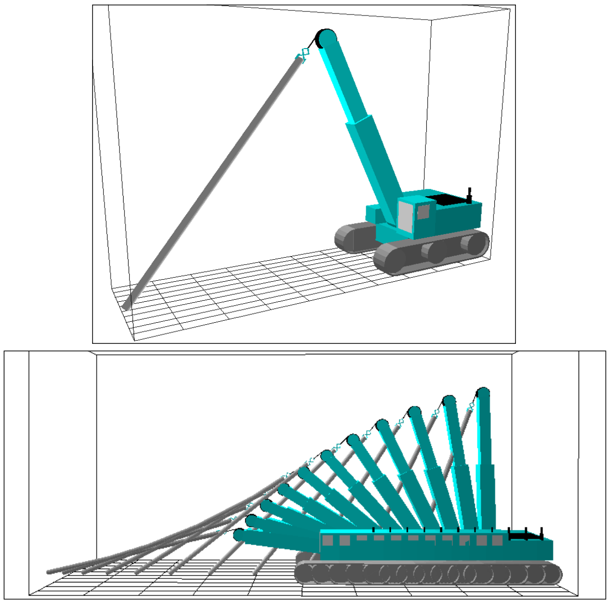

Figure 6: L'extrimité basse du tube est continuellement en contact avec le sol lors du déplacement de l'engin.

Les méthodes probabilistes sont utilisées pour la construction de ce graphe. La contrainte spécifique au problème de manipulation est prise en compte dans la méthode locale utilisée. La méthode locale tient en compte la présence d'une chaîne cinématique fermée.

La figure 7 présente un exemple de grue télescopique manipulant un objet encombrant. Bien que l'espace de configuration du manipulateur seul est connexe, $GRASP \cap PLACEMENT$ ne l'est pas. On distingue dans la figure 7 les deux composantes connexes capturant la topologie de cet espace.

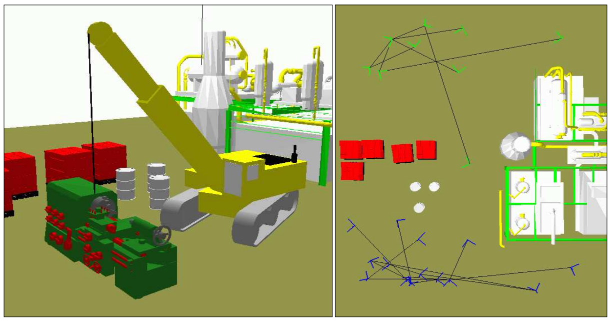

Figure 7: Graphe calculé dans $GRASP \cap PLACEMENT$.

Afin de calculer un graphe de manipulation, on opère en plusieurs phases. Au début, on calcule les composantes connexes de l'espace $GRASP \cap PLACEMENT$ en utilisant l'approche décrite ci-dessus. On détermine après, la connectivité des composantes connexes de ce graphe avec des chemins de transfert et de transit calculés en s'appuyant sur l'approche qui combine visibilité PRM et RRT décrite dans la section 5. Finalement pour trouver un chemin dans l'une des composantes connexes de $GRASP \cap PLACEMENT$, on utilise le graphe calculé par la méthode utilisant les chaînes fermées décrite ci-dessus.
L'intérêt de cette approche est d'étendre la notion d'ensemble discret de placements des objets déplaçables adoptée par les autres approches à un ensemble continu.

8 Conclusion

Nous avons présenté dans cet article plusieurs approches pour résoudre des problèmes de planification de tâches de manipulation. Le graphe de manipulation dans la dernière approche n'est pas explicitement calculé. La recherche d'un chemin de manipulation aboutie si les deux méthodes utilisés dans la dernière approche, réussissent à trouver un chemin. Une analyse théorique est en cours d'étude afin de prouver la complétude de cet algorithme.
Nos travaux futurs se consacreront à l'extenssion au cas d'ensemble de prise continu des approches décrites dans cet article.

Références

[1] J.-M. Ahuactzin, K. Gupta, and E. Mazer. Manipulation task planning for redundant robots: A practical approach. *International Journal of*

Robotics Research, MIT Press, pages 731–741, 1998.

[2] R. Alami, J.P. Laumond, and T. Siméon. Two manipulation planning algorithms. *The first Workshop on the Algorithmic Foundations of robotics.(WAFR'94),San Fransisco*, pages 109–125, February 1994.

[3] R. Alami, T. Siméon, and J-P.Laumond. A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps. *In Robotics Research: The Fifth International Symposium,Tokyo*, August 1989.

[4] P. Bessiere, J.M. Ahuactzin, E.-G. Talbi, and E. Mazer. The ariadne's clew algorithm: Global planning with local methods. *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2:1373–1380, 1993.

[5] L. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Int'l Conf. on Robotics and Automation (ICRA'1996)*, pages 566–580, 1996.

[6] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. *IEEE International Conference on Robotics and Automation*, 2:945–952, 1994.

[7] J.J. Kuffner and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. *IEEE Int'l Conf. on Robotics and Automation (ICRA'2000),San Francisco*, 2000.

[8] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Technical Report No. 98-11, Dept. of Computer Science, Iowa State University*, 1998.

[9] C.L. Nielsen and L.E. Kavraki. A two level fuzzy prm for manipulation planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[10] C. Nissoux, T. Simeon, and J.-P. Laumond. Visibility based probabilistic roadmaps. *IEEE International Conference on Intelligent Robots and Systems*, pages 1316–1321, 1999.

[11] T. Simeon, J.-P. Laumond, and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics Journal*, 2000.